smsqzine

Issue #3 June 2016

SMSQzine table of contents

Published by:
Timothy Swenson
swenson_t@sbcglobal.net
swensont@lanset.com

SMSQzine is published as a service to the Sinclair QL community. Writers are invited to submit articles for publication. Readers are invited to submit article ideas.

Created using Open Source Tools:

- OpenOffice
- Scribus
- Gimp
- SMSQmulator

Editorial **1**

The QL in the United States **1**

Astronomical Calculations with Abacus and Archive **3**

Random Numbers Across Emulators **4**

Sigot Algorithm for PI **5**

Quality of QL Random Number Generator **6**

Prospero Fortran **7**

Editorial

It's been a while since the last issue was out, but I've had a number of distractions from the QL. The articles focus on those things that interest me, leaning toward programming and computation. I do have problems coming up with ideas for articles, so if you have ideas, please pass them along.

The cover is from the first Sinclair QL advertisement package that I received back when the QL was released to the US. I remember going over it in detail, marveling at the graphics, the four application programs that came with the QL, wondering what the QL Net network really was, and so on. I was impressed at how black the black was in the monitors. I remember seeing C64's with their composite monitors and the black would get brighter as you turn up the brightness. That never happened with the QL Vision monitor. I remember realizing that the foldout picture of the QL was life size, so I would set it down and see how the spacing on the keys were by practicing typing on the picture. I kept that advertisement package all these years and even picked up a few more copies. It's has been 30 years since I bought my first QL, in April 1986, back when I was on college. Now, my middle daughter is now graduating from the same college.

The QL in the United States

When the Sinclair QL was launched in the UK, the American computer press picked up the news. Infoworld, a leading computer news magazine, announced the launch in its Feb. 6, 1984, issue. In the next issue of Infoworld it said that Sir Clive announced that the QL will be sold in the United Status starting in the fall of 1984. The expected price will be $499. The plan was to sell the QL via mail-order.

In February of 1984, Timex Computer Corporation threw in the towel and stopped making and selling the Timex/Sinclair 1000, 1500, and 2068. The home computer wars were heating up and Timex was one of the casualties. Texas Instruments announced in October 1983 that they were discontinuing the TI 99/4A. Also in October, Mattel announced its discontinuance of the Aquarius home computer, less than 6 months after it was released.

Throughout all of 1984, Sinclair was having problems with the QL in the UK market. The effort to get into the US market was delayed. Sinclair employee David Chatten found the Korean manufacturer, Samsung, to make the US and German QL's. Since the RF standards in both countries were tighter than the UK, the Samsung QL's were

First QL Only Newsletter in the US

electronically quieter and better built. John Munford managed the relationship with Samsung, making many trips to Korea.

David Ahl, founder and editor of Creative Computing, one of the early computer magazines, wsa able to get a hold of a QL. The December 1984 issue of Creative Computing had an extensive review of the QL, written by David himself. David found a few problems with the keyboard and the microdrives. He was impressed with the manual. One of the more interesting parts of the review is his comment on Easel: "On the other hand, the QL Easel business graphics package is outstanding, and it alone may justify the purchase of the system, especially for a business which already has an Epson FX80 (which prints the graphics output exactly)."

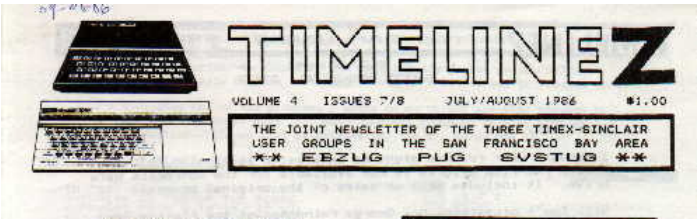
In February, 1985, after being the Managing Director of Sinclair Research, Nigel Searle was shipped back to the United States to handle the launch of the QL. He had previously handled the launch of the ZX81 in the United States. The launch of the QL was set for June 1985.

As the date of the launch for the QL was coming up, the financial fortune of Sinclair Research was on the decline. Both Sinclair and Amstrad had price cuts early in 1985. It was estimated that Sinclair Research was losing a million pounds a month at the start of 1985 and by the middle of 1985, it had 30 millions pounds of unsold inventory. With bugs in the QL getting sorted out, Sinclair decided to relaunch the QL in March 1985 with a 750,000 pound advertising campaign. In May 1985, Sinclair admitted that it was in dire financial straights, despite the sales numbers increasing after the second launch.

The QL was launched in May 1985 with the first sales going to American Express customers. The launch was very subdued and I could not find any specific mention of the launch. Infoworld, which had been reporting on the QL, did not report the launch.

Timelinez, the newsletter for the San Fransicco Timex Sinclair User Groups, did mention that they demonstrated the QL at the West Coast Computer Faire, in March 1985. The user groups had a booth at the Faire and handed out 1,750 QL sales brochures. The QL, a UK model, was on loan from Sinclair USA, arranged by Mary Reinman.

Orders for the general public were taken, starting in June 1985. It was reported that those that ordered directly from Sinclair received their QL's before those ordered by American Express customers.


TIMELINEZ

VOLUME 4 ISSUES 7/8 JULY/AUGUST 1986 $1.00

THE JOINT NEWSLETTER OF THE THREE TIMEX-SINCLAIR USER GROUPS IN THE SAN FRANCISCO BAY AREA

EBZUG PUG SVSTUG

Norm Lehfeldt of San Francisco reported that it took 5 months from order for his QL to arrive, in September 1985.

In August 1985, the cost of the QL was reduced from $500 to $300. As 1986 came around, the financial situation for Sinclair was looking worse. In April, 1986, Sinclair was sold to Amstrad. Before this happened, George and Carol Whitham of A+ Computer Response, worked out a deal with Sinclair

USA to purchase all of the Sinclair stock in the United States, dealing with Terry Shurwood, director of Sinclair USA. George and Carol initially approach Sinclair USA to be a reseller.

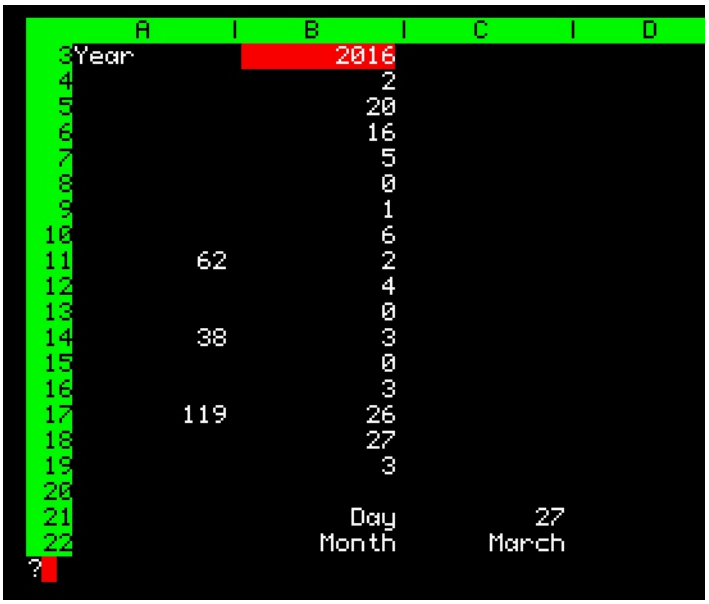
The QL really did not make it into any major computer or electronics retailers. A number of small computer stores, those that already sold Timex/Sinclair computers, were the only retailers to carry the QL. A number of mail order companies, like Bob Dyl's QL Connection, Curry Computer of Arizona, Mechanical Affinity run by Frank Davis and Paul Holmgren, and Quantum Computing. The QL Connection advertised in Infoworld and Quantum Computing advertised in Dr. Dobb's Journal, both leading computer magazines. Andy Hredesky said that he sold the QL going door-to-door in Sunnyvale, California.

It was the numerous Timex/Sinclair user groups that provided a market for the QL. Most QL owners had previously owned a Timex/Sinclair computer. As the first QL's started showing up at the user group meetings, more were interested in getting their own.

Initially there was no magazine for the QL in the United States. Users used the T/S user group

newsletters and T/S magazines like Time Designs and Update!. The first QL-only newsletter was probably Quantum Levels, published by Tom Bent. Soon after was QLui, which was a very short lived newsletter that I believe only had two issues. The largest QL-only newsletter was International QL Report (IQLR) with its first issue in May/June 1991.

As a ZX81 and T/S 2068 user, and an undergraduate in Computer Science, I was watching the press on the QL. I did not have any plans to buy right away. The $500 cost was more than the cost of a college Quarter, both tuition and books. I needed more time to save my money. In April, 1986, I visited Sunet Electronics in San Francisco. The QL was down to $300, the QL Vision monitor was $300, and the QL printer was $300. If you bought all three in a bundle, the cost was $800. I walked out the door with the whole bundle. Soon, I was typing my school papers on the QL and using the near-letter-quality of the printer to print them out.



Calculating Easter with Abacus

Astronomical Calculations with Abacus and Archive

The books that I found on Astronomical Calculations mostly talked about calculators, with some using programming languages to implement the calculations or algorithms. The older books used BASIC or Pascal, some newer books used C or C++.

One book that was interesting was "Practical Astronomy with our Calculator or Spreadsheet" by Duffet-Smith & Zwart. This book was an updated version that talked just about using a calculator and added the idea of using a spreadsheet for the calculations. I really had not though about using a spreadsheet and I used to do a lot of spreadsheets with Lotus 1-2-3, back in the day. I decided to see if Abacus would be up to the task for some of the calculations.

A number of the simpler calculations were fairly easy to implement in Abacus. The more advanced calculations in the book relied on the ability of MS Excel to create new functions, which is something that Abacus is not able to do. I did have to be careful about the recalculation method in Abacus and make sure that I did not have cells that needed to be calculated first, being calculated second.

The algorithms that I was able to implement in Abacus are; Easter, Day of the Week, Moon phase, Gregorian to Julian and Julian to Gregorian. They are part of the separate download as the files:

easter_aba
dayofweek_aba
Greg2jul_aba
jul2greg_aba
moonphase_aba

After implementing these calculations with Abacus, I was wondering what other tools or languages that I could use to. One of the other parts of Xchange is Archive. Known as a database package, Archive has a programming language that does have the essential functions needed to implement these calculations. The resultant code is very similar to Superbasic, so it was fairly easy to port over. The algorithms that were implemented are; Easter, Moon phase (which includes Gregorian to Julian) and Day of the Week. They are part of the separate download as the files:

Easter_prg
moonphse_prg
dayofweek_prg

Example Code: Day of the Week

```
proc greg2jd
   if month = 1 or month = 2
   let year = year - 1
   let month = month + 12
   endif
   let a = int(year/100)
   let b = 2 - a + int(a/4)
   let jd = int(365.25*(year+4716))
   let jd = jd +
int(30.6001*(month+1))
   let jd = jd + day + b - 1524.5
endproc
proc dayofweek
   input "Year  : ";year
   input "Month : ";month
   input "Day   : ";day
   print
   print "Day is ";
   greg2jd
   let jd = jd + 1.5
   let x = jd - (7 * int(jd/7))
   if x = 0
   print "Sunday"
   endif
   if x = 1
   print "Monday"
   endif
   if x = 2
   print "Tuesday"
   endif
   if x = 3
   print "Wednesday"
   endif
   if x = 4
   print "Thursday"
   endif
   if x = 5
   print "Friday"
   endif
   if x = 6
   print "Saturday"
   endif
endproc
```

Random Numbers across Emulators

Years ago I wrote a program called Complex Ascii Rotation (CAR). Ascii rotation is taking a character and adding a set value to it. ROT-13 was a common rotation used in USENET forums to "hide" some material that some might find offensive. ROT-13 was taking a character and adding 13 to its value. In the case of an A, ROT-13 turns it into N, B becomes an O and so on.

Technically this is a form of encryption, but it is too weak and obvious to be secure. For each character the rotation is the same, so with just a few example letters, it is fairly easy to discover how much each letter is being shifted.

If the value shifted was different for each letter, that would complicate the encryption. The problem is determining the number to shift each letter. One needs a list of numbers for this. This is where the random number generator comes in. Most home computers had a pseudo-random number generator that used an algorithm that would create a list of the same numbers when starting with the same starting point. This is where the RANDOMISE command comes in. For the ZX81, it took the FRAMES value as the seed for the random number generator. On the QL, it worked similarly.

This all assumes that you want random numbers. But if you seed the random number generator with the same number, then you will get the same list of numbers every time. Complex Ascii Rotation used this flaw in the system to find a different value to shift each letter. A password from the user was hashed into a single number. This was then used as the seed for the random number generator. The program would then read in a character, get a "random" number, add that value to the character and output that number to a file. In this way, the entire original message file was processed. The ending file would look like gibberish.

Since the list of random numbers depended on the home computer used, it was fairly secure. You could

not "encrypt" on the QL and then "decrypt" on a Spectrum or C64. The receiver had to have the same computer as the sender.

As the QL expanded into different emulators and OS versions (Minerva, SMSQ/E), I wondered if there was any incompatabilites between them that could cause Complex Ascii Rotation to fail. I decided to run a simple test on a couple of emulators and different ROM's.

Here is 10 random numbers generated using the same starting point, on different emulators and ROMs:

QDOS - Q-emulator with JS ROM
5, 3, 1, 10, 5, 0, 1, 8, 8, 9
Minerva - Q-emulator with Minerva 1.97 ROM
5, 3, 1, 10, 5, 0, 1, 8, 8, 9
SMSQ/E - Q-emulator with Gold Card SMSQ/E
5, 5, 9, 2, 7, 0, 1, 2, 8, 5
SMSQ/E - QPC 1
5, 5, 9, 2, 7, 0, 1, 2, 8, 5
SMSQ/E - QPC II
5, 5, 9, 2, 7, 0, 1, 2, 8, 5
SMSQmulator
5, 5, 9, 2, 7, 0, 1, 2, 8, 5

The conclusion is that both QDOS and Minerva use the same random number generator. SMSQ/E (on any platform) uses the same random number generator, but different than QDOS and Minerva.

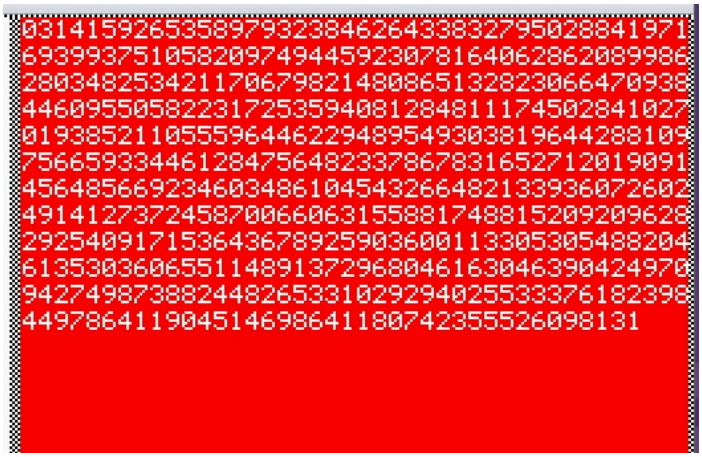
The code used to run the test is:

```
10 randomise 100
20 for x = 1 to 10
30    print rnd(10)
40 next x
```

Spigot Algorithm for PI

While trolling the Internet for something else, I came across an algorithm for generating N digits of PI. The page (http://rosettacode.org/wiki/Pi) is part of the website Rosetta Code, that shows different coding projects in a number of languages.

The pi algorithm is called a Spigot algorithm because "it pumps out digits one at a time and does not use the digits after they are computed." The algorithm was published by Stanley Rabinowitz and Stan



Wagon in 1995. Their algorithm was based on a spigot algorithm for the value e (lower case E) created by A. H. Sale in 1968.

I'm not a mathematician, but what interested me was that the algorithm does not require any floating point math, but only integer math. It also does not require the handling of large numbers. The computation involved is fairly simple. The only down side of the algorithm is that, the more digits you want to compute, the more memory is needed. The higher the number of digits, the longer the processing is. Computing the first 10 digits when computing 10 digits is quick. Getting the first 10 digits when computing for 10,000 digits is a lot slower. The output does not generate the decimal place for PI, meaning that the output starts with 0314159..., which is really 3.14159....

From the Rosetta Code website, I found the link to the original article from 1995. It had an implementation of the algorithm in Pascal. The code worked fine with Computer One Pascal. There was a C version that compiled fine with C68.

The website has a version of algorithm for BASIC that was fairly easy to convert the SuperBasic. The BASIC version was for BASIC256, a version of BASIC that is very similar to SuperBasic, when compared to QBASIC or other popular forms of

BASIC. This version was fairly easy to port to SuperBasic and get running on the QL.

With the SuperBasic version I ran it to calculate PI to 500 digits (see diagram).

If one needed to calculate and then store the first X digits of PI, the code would be fairly simple to add saving the digits to a file. Just remember that the more digits calculated, the longer it will take to run through the process.

Structured SuperBASIC version of Algorithm

```
## Pi Spigot;

   n = 100
   length = (10*n) / 4
   dim a(length)

   for j = 1 to length
      a(j) = 2
   next j
   nines = 0
   predigit = 0
   for j = 1 to n
      q = 0
      for i = length to 1 step -1
         x = 10*a(i) + q*i
         a(i) = x mod (2*i-1)
         q = int(x / (2*i-1))
      next i
      a(1) = q mod 10
      q = int(q / 10)
      if q = 9 then
         nines = nines + 1
      else if q = 10 then
         print predigit+1;
         if nines > 0 then
            for k = 1 to nines
               print 0
            next k
         end if
         predigit = 0
         nines = 0
         else
            print predigit;
            predigit = q
            if nines <> 0 then
               for k = 1 to nines
                  print 9;
               next k
               nines = 0
            end if
         end if
      end if
   next j
   print predigit;
```

Quality of QL Random Number Generator

After looking at the QL random number generator across emulators, I started thinking about the quality of the random number generator. Is the random number generator on the QL with SMSQ/E good or not so good?

I wrote a short program to generate a number of random numbers from 1 to 10, then for each random number, I incremented the value in an array to keep track of how many times that number appeared. I ran this program for a hundred, a thousand, and ten thousand iterations. The results were entered in to an Abacus spreadsheet and a little computation was made on the numbers. Mostly the maximum, minimum and difference between the two was calculated (see Figure 1). This give an idea of how random the random number generator is.

To look for a different or better random number generator I found the book "Numerical Recipes in C, 2nd Edition". Chapter 7 is dedicated to random numbers and it presented a couple of random number algorithms, some fast and some good. The good random number generators are listed as ran0, ran1, and ran2. These algorithms are designed to be portable across systems and languages. Ran0 is the Minimal Standard generator proposed by Park and Miller. It is a decent generator, but to does have some flaws. The authors of the book created ran1 to fix the deficiencies of ran0. They say that "ran1 passes those statistical tests that ran0 are known to fail." The other random number generator, ran2, has a longer period than ran1 and should only be used

when one needs a long sequence of random numbers.

I ported to the ran1 algorithm to SuperBasic and run it for a hundred, a thousand, and ten thousand iterations. I then plugged the data into a similar spreadsheet as the QL random number generator (see figure 2).

Looking at the two spreadsheets, there is not a whole lot of difference between the random number generators. The ran1 algorithm seems to be a little less well distributed at one hundred iterations, but at ten thousand iterations, it comes pretty close to the QL random number generator. If ran1 is considered by the authors of "Numerical Recipes in C, 2nd Edition" and it is comparable to the QL SMSQ/E random number generator, then the QL random number generator must be considered good, too.

	A	B	C	D	E	F
1	QL RND	100	1,000	10,000	100,000	
2						
3	1	10	104	1038	9952	
4	2	11	95	994	10081	
5	3	9	110	1038	10070	
6	4	10	104	1002	9964	
7	5	9	101	986	10040	
8	6	10	112	965	9853	
9	7	9	93	972	9881	
10	8	11	100	1033	10043	
11	9	12	103	972	10256	
12	10	9	78	1000	9860	
13		-----	-----	-----	-----	
14	Sum	100	1000	10000	100000	
15	Mean Avg	10	100	1000	10000	
16	Max	12	112	1038	10256	
17	Min	9	78	965	9853	
18		3	34	73	403	
19						
20						
?						

Prospero Fotran

The Prospero Fotran compiler and manual was made available a while back. I'm not a big fan of Fortran, but it was a class that I took in college. As a compiler for the QL, I wanted to give it a try.

The Fortran compiler is similar to the Prospero Pascal compiler in how it comes and how it works. The compiler is partially stored on a ROM cartridge. A software of the cartridge comes with the compiler, so you will need an emulator that allows ROM cartridges, which means Q-Emulator. The compiler also comes with an extension toolkit, but this is only needed for running the Fortran programs on computers without the ROM. Don't have the ROM loaded and the load the toolkit. The compiler will work once and then give an error the next time.

The compiler documentation is pretty good and about the same as the Prospero Pascal compiler. The manual comes in three parts, an overview, a review of the language, and using the compiler.

With a little hope from Urs Konig, I figured out to use the compiler. It is a two step process - run the compiler and then run the linker. The end result with be a _BIN file with is the QL executable.

	A	B	C	D	E	F
1	RAN1	100	1,000	10,000	100,000	
2						
3	1	12	100	1041	10237	
4	2	10	103	991	9938	
5	3	11	91	983	10100	
6	4	9	108	1011	10119	
7	5	11	108	990	9978	
8	6	11	93	978	9862	
9	7	5	115	1006	9911	
10	8	4	88	991	9840	
11	9	18	103	995	9885	
12	10	9	91	1014	10130	
13		-----	-----	-----	-----	
14	Sum	100	1000	10000	100000	
15	Mean Avg	10	100	1000	10000	
16	Max	18	115	1041	10237	
17	Min	4	88	978	9840	
18		14	27	63	397	
19						
20						
?						

The compiler comes with three example Fortran programs; prime, squares, and trapdemo. Prime is a program that determines if a number is prime, squares is a demonstration of the QDOS graphics routines, and trapdemo is a demonstration of using QDOS traps from Fortran.

The compiler supports Fortran 77, which at the time the compiler came out, was the most recent version of Fortran. To test if the compiler would handle standard Fortran 77, I

found a Fortran 77 book and picked a piece of sample code and ran it through the compiler. The sample code did leave out the declaration of the variables, but other than that, I made no other changes to the code. The code compiled with no errors.

When looking for code to try I found a problem with Fortran 77 and what is called Fortran 77. The Fortran compiler that comes with Linux is f77, or a Fortran 77 compiler. But it was updated after the next version of Fortran (Fortran 90) came out, so it picked up some modifications that are not port of the Fortran 77 standard. One sign that a Fortran 77 program is updated is it using lower case. Fortran 77 stayed with the older convention of user all uppercase. Another indicator is using "end do" instead of "CONTINUE" when ending a DO loop. When buying books on Fortran 77, I had to make sure was printed before 1990.

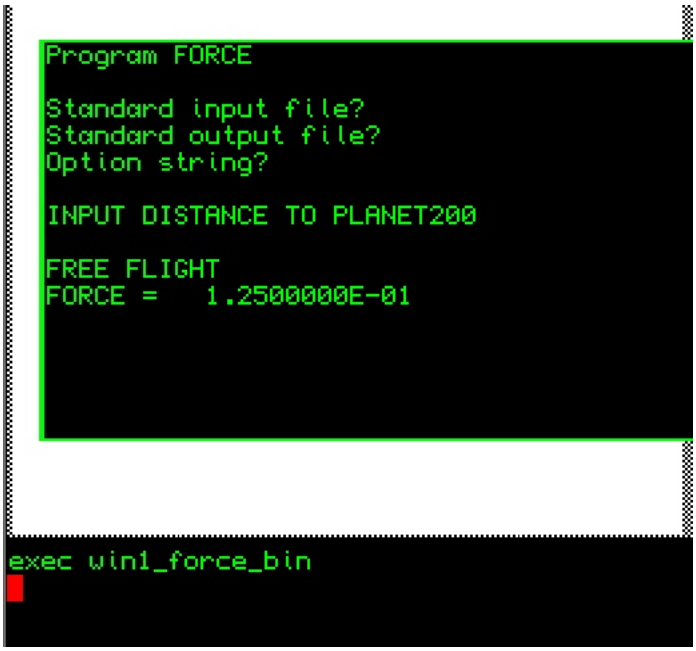
Like the Prospero Pascal, Prospero Fortran only allows windows to be opened for output and no input. This is probably the biggest issue when it comes to creating QL programs in Fortran.

When running the compiler programs, there is a standard three questions that come up first:

Standard input file?
Standard output file?
Option string?

These questions can be annoying. There is no way to turn these question off in the compiler or linker, but there is an accessory program (noqns_exe) that can be run on the compiled code to turn this off the questions.

Granted Fortran is not one of the popular languages, put there is a large amount of available source code, esp. in mathematical and scientific programming. I can see using the compiler to port over older Fortran 77 programs and try them out on the QL.




```
      PROGRAM FORCE
         REAL  D,F
         PRINT *,'INPUT DISTANCE TO
PLANET'
         READ *,D
         IF (D .LE. 100.0) GO TO 10
         F = 5000.0/D**2
         PRINT *,'FREE FLIGHT'
         GO TO 20
10       F = 5000.0/D**2-8.0
         PRINT *,'THRUSTERS ON'
20       PRINT *,'FORCE = ',F
      END
```